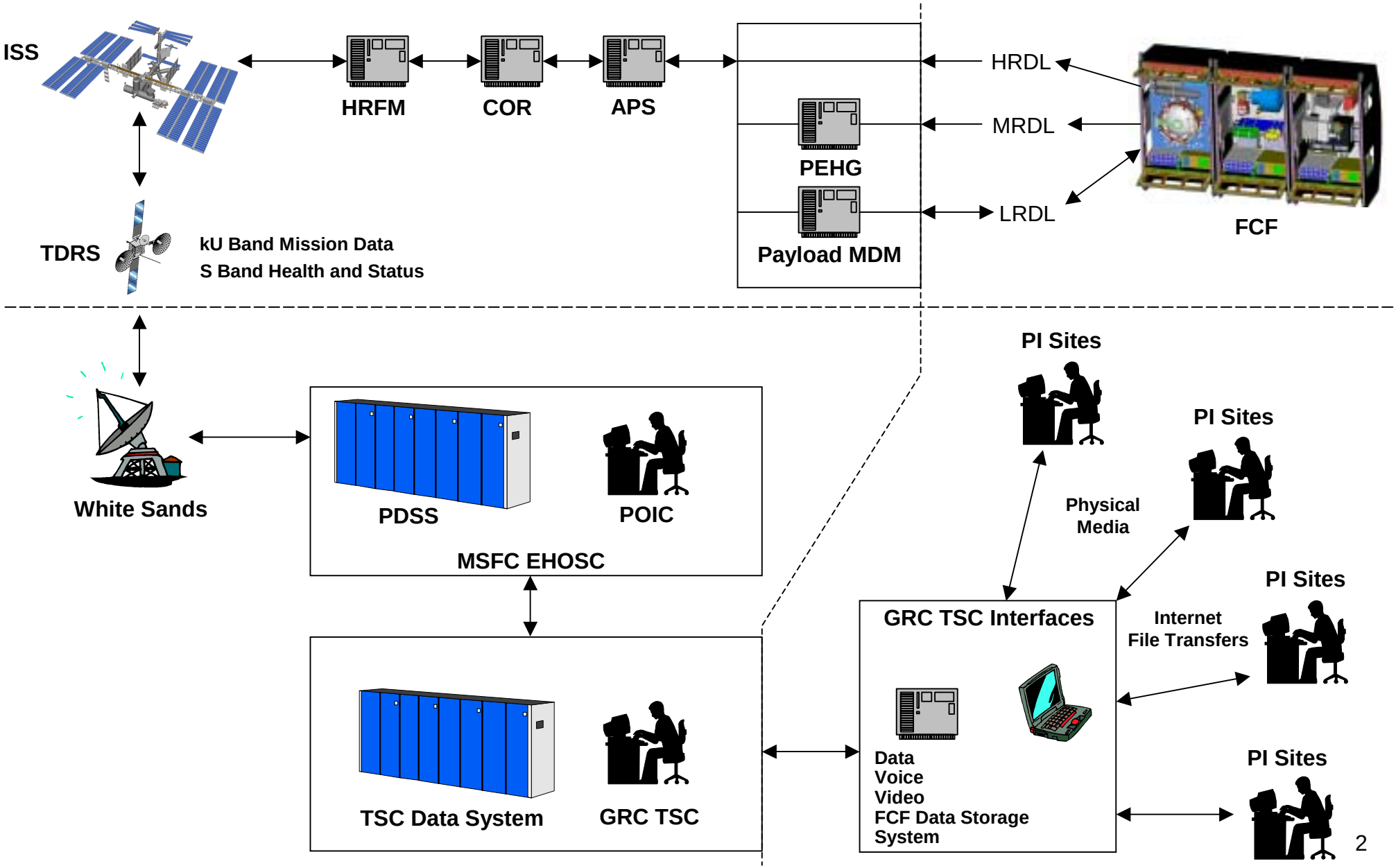
# FCF Software

Stephen Lux

February 13, 2001

# FCF Software Data Flow



# FCF Software Development Methodology

## Software Spiral Lifecycle

- A phased approach to developing FCF Flight Software
- More accurately reflects what is really done
- Requirements and design evolve as more is learned from prototyping
- Allows for alternatives to be explored during software development
- Allows for rapid prototyping

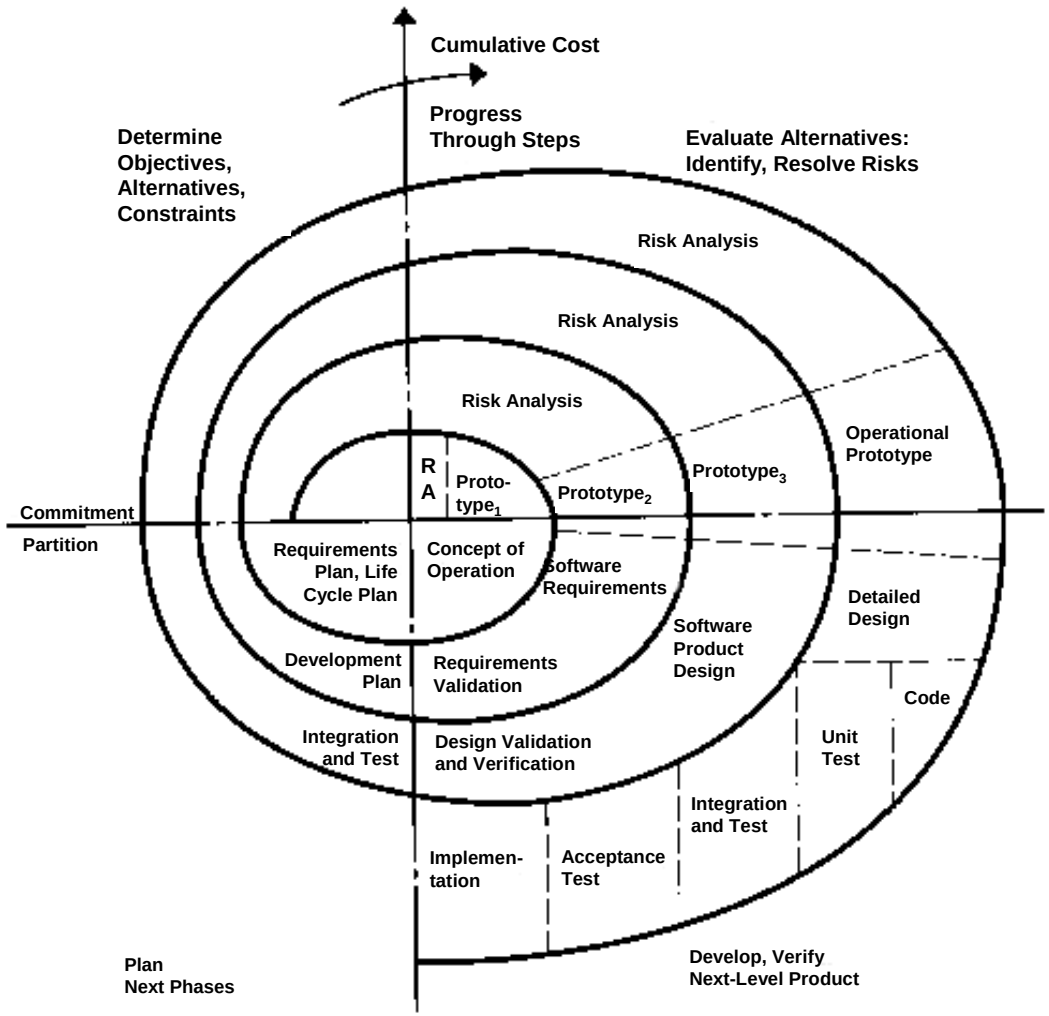



Diagram from “Managing the Software Process” by Watts S. Humphrey of the Software Engineering Institute (SEI) at Carnegie Mellon University

# FCF Software Life Cycle Considerations

- On-Orbit Flight Segment software will be modifiable by the ground through the communication network available to International Space Station (ISS) payloads
- Flight and Ground Software will be modularized so that mission specific code sequences and parameters can be changed without reloading non-mission specific code, sequences, and parameters
- After deployment of the Shared Accommodations Rack (SAR), the Flight Segment will have the capability to transfer functions from the primary processor to at least one other processor
- Flight Software is designed to accommodate new hardware and software configurations (i.e. new Principal Investigator (PI) Hardware and Software)
- Goal is to minimize the amount of new software required for a new configuration
- Autonomous Operation of Facility (emphasis on Ground Control of FCF)

# Software Requirements for FCF Architecture

**Primary List of Parent Documentation**

- Pressurized Payloads Interface Requirements Document (SSP 57000)
- Pressurized Payloads Software Interface Control Document(SSP 57002)
- Pressurized Payloads Accommodations Handbook (SSP 57020)
- ISS Computer-Based Control System Safety Requirements (SSP 50038)
- FCF to ISS Software Interface Control Document (SSP 57319)
- Safety Policy and Requirements for Payloads Using the International Space Station (NSTS 1700.7B, ISS Addendum)
- System Specification for the Fluids and Combustion Facility (FCF-SPC-0001)
- Prime Item Development Spec: Combustion Integrated Rack (FCF-SPC-0002)
- Prime Item Development Spec: Fluids Integrated Rack (FCF-SPC-0003)
- Prime Item Development Spec: Shared Accommodations Rack (FCF-SPC-0004)
- Science Requirements Envelope Document (SRED) (FCF-DOC-0002)
- FCF Baseline System Description (FCF-DOC-003C)
- Applicable FCF Hardware Level C Specifications

Note: Because of the large number of requirements and the complexity of the flow down to lower levels, the Dynamic Object Oriented Requirements System (DOORS) software was purchased. This program allows for traceability to requirements among multiple documents and allows for customization to accommodate detailed compliance descriptions.

# Key Software Design Drivers

- Flight Software is designed to accommodate new hardware and software configurations (i.e. new PI Hardware and Software or Orbital Replacement Unit (ORU) Upgrades)
- Goal is to minimize the amount of new software required for a new configuration
- All new Flight Software Loads will need to be verified and validated prior to uplinking to the FCF
- Several different systems require software:
  - Flight Unit (FU)
  - Experiment Development Unit (EDU)
  - Ground Integration Unit (GIU)
  - Payload Training Center Unit (PTCU)
  - Ground Support Equipment (GSE)
  - Simulators
- Utilize latest software development tools and techniques and implement an Object Oriented Analysis and Design philosophy to maximize software commonality, reusability, and maintainability
- Flight Software is designed to accommodate hardware resource sharing between racks

# Object Oriented Analysis, Design and Software Reuse for FCF
## (Common FCF Base Class Framework)

**Base Classes are formed at the FCF Level**

- Using Inheritance and Polymorphism, derived Classes are formed at the Package Level
- Derived Classes are specific to the Package and Hardware
- The Base Classes are reused, promoting better design, more thorough testing, and thus fewer defects at the Package Level

**Analyze Common Physical Components**

- Similar Hardware Components
- Extract Common Attributes or Properties (Data)
- Extract Common Processes or Methods (Functions)
- Create the Base Class that encapsulates Attributes and Processes, using Unified Modeling Language (UML) captured via the Artisan Real-Time Studio Software

# Object Oriented Analysis, Design and Software Reuse for FCF – Continued

**(Common FCF Base Class Framework)**

**Analysis is done using the Problem Domain**

- Physical Objects are analyzed
- Identify the Properties of the Objects
- Identify the Processes that operate on the Properties and the Processes required for interaction with other Objects (messages or communication)

**Other Advantages of the FCF Base Class Framework**

- Framework provides the template for package designs, promotes a common method for all packages
- Shorter Design and Code Times due to reuse of Base Class
- When complete, the FCF Base Class Framework represents an “Off the Shelf Reusable library” for future projects
- Will encourage a consistent Application Program Interface (API)

# Object Oriented Analysis, Design and Software Reuse for FCF – Continued

## (Common FCF Base Class Framework)

**Potential Examples of FCF Base Classes**

- FCF Application Object
- Process includes POST (Power On Self Test)
- Encapsulates Application Global Data (Static)
- CAN Bus Controller
- Generic Motor Controller
- A/D, D/A
- Frame Grabber
- RS 422
- Ethernet
- Package Communication Interfaces
- Common C++ Classes (String, Lists, etc…)

**Use of Common FCF Base Class Summary**

- Better Design
- More Thorough Testing
- Fewer Defects
- Shorter Schedule
- Base Class Frameworks when complete represents a reusable library for future projects

# FCF Software Development Tools

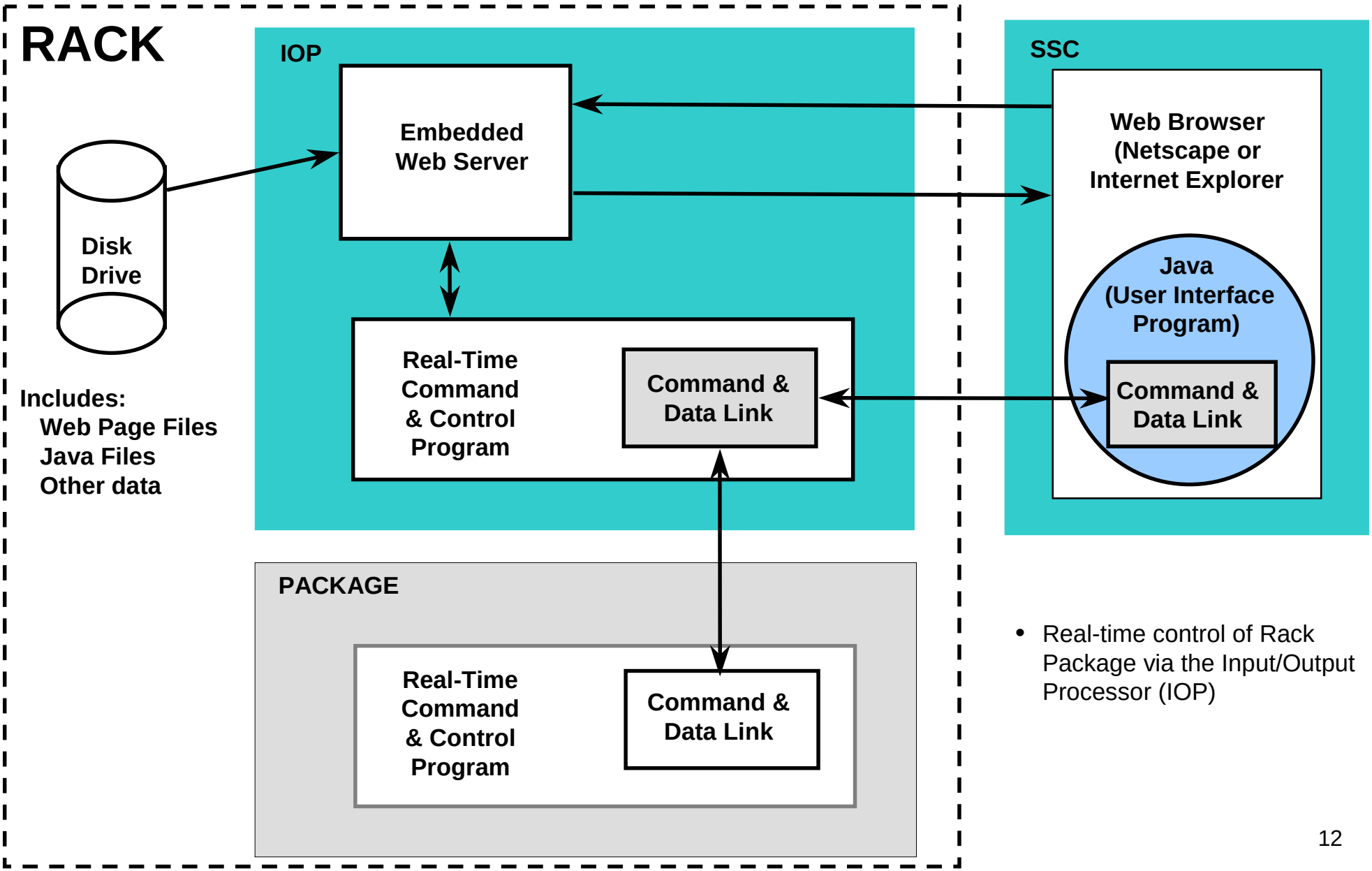
| Description | Tool | Status |
|---|---|---|
| Operating System | VxWorks | Selected |
| Languages (Embedded) | C++ | Selected |
| Language (GUI) | Java | Selected |
| Communications | CAN Bus & Ethernet | Selected |
| Embedded Web Technology | Wind River Systems Web | Selected |
| C++ Compiler | GNU | Selected |
| VxWorks Development Environment | Tornado | Selected |
| Software Design Tool | Unified Modeling Language (UML) | Selected |
| Configuration Management Tool | PVCS | Selected |
| Software Test Tools | Bus Analyzers, Code Analyzers, Package Simulators | Under Review |
| Requirements Management | DOORS | Selected |

# Embedded Web Technology (EWT)

EWT allows a user with a computer and Web browser to monitor and/or control a remote device with an embedded system using a convenient, graphical user interface.

- World Wide Web standards – Hyper Text Transfer Protocol (HTTP), Hyper Text Markup Language (HTML)
- Java
- Only a browser required to run the user interface software
- Cost savings through software reuse and portability
- Allows FCF to take advantage of work done by others
- Using commercial standards makes it easier to hire trained personnel
- Simplifies the development and maintenance of FCF Software
- No need to develop custom protocols
- Allows for simpler upgrades and expansion in the future
- Simplifies the incorporation of flight displays into ground operations

# EWT - SSC/IOP/Package Interface




- Real-time control of Rack Package via the Input/Output Processor (IOP)

# Capability Maturity Model (CMM) Level 3

- CMM was developed by the Software Engineering Institute (SEI) located at Carnegie Mellon University
- Model for Software Organization and Process Improvement
- Key Process Areas are identified at each CMM Level
  - Level 1 (Initial)
  - Level 2 (Repeatable)
    - Requirements Management
    - Software Project Planning
    - Software Project Tracking & Oversight
    - Software Subcontract Management
    - Software Quality Assurance
    - Software Configuration Management
  - Level 3 (Defined)
    - Organization Process Focus
    - Organization Process Definition
    - Training Program
    - Integrated Software Management
    - Software Product Engineering
    - Inter-group Coordination
    - Peer Reviews

# Testing Methodology

**Based on Capability Maturity Model (CMM) (Level 3)**

- **Unit Testing (Computer Software Unit)**
  - Unit testing is conducted throughout the implementation process, first as part of the initial development process and later as changes to the unit are made
  - Unit tests are repeatable and may be conducted at any point in the implementation process in accordance with the approved unit test plan
  - The goal for unit testing by developers is to perform selected path testing in which every affected branch is navigated in all possible directions at least once and every affected line of code is executed at least once
  - Unit test drivers and stubs will be developed as needed and will be placed under configuration control as part of the overall test utility
- **Unit Integration and Testing (Computer Software Component)**
  - At this level, Computer Software Units (CSU) are incrementally integrated to form continually larger and more complex software builds
  - The purpose of this level of testing is to both identify errors and demonstrate interface compatibility
  - Integration continues until all software Configuration Items are integrated with the system-level hardware suite into a single functioning software system

# Testing Methodology – Continued

- **Computer Software Configuration Item (CSCI) Qualification Testing**
  - The purpose of CSCI Qualification Testing is to verify satisfaction of CSCI performance requirements documented in the Software Requirement Specifications.
  - Computer Software Configuration Item Qualification Testing for the FCF Project will take place only on the target computer system, interfaced with hardware and software components of the software test environment. This restriction will assure that testing of timing, capacity, throughput, and responsiveness of FCF Project CSCI components with respect to performance requirements can be accurately assessed.
- **CSCI/HWCI Integration and Testing**
  - Upon completion of the testing and integration of all builds and the preliminary integration testing of all Computer Software Configuration Item (CSCI) components comprising the FCF Project, the software is ready for CSCI/HWCI testing. (HWCI – Hardware Configuration Item)
  - The objective is to validate that the FCF Project Hardware and Software components can individually be interfaced in accordance with the Software Requirements Specification (SRS) and Interface requirements and that the components taken as a system are stable enough to proceed with System Qualification Testing.

# Testing Methodology – Continued

- **System Qualification Testing (SQT)**
  - FCF Project System Qualification Testing consists of complementary and progressive test phases. Once CSCI/HWCI is completed by the FCF System Test Organization, System Qualification Testing (SQT) begins.
  - A single Software Test Plan will be generated to address the planning for all levels of software SQT. A Software Test Document (STD) will be generated for each CSCI component, documenting the test procedures to be run to verify each requirement in the SRS for that component. A Software Test Report (STR) will be generated for each CSCI component, documenting the results of each CSCI component test.
  - A cross reference matrix will be provided, using the project wide requirements traceability database, to document the test or tests that satisfy each SRS requirement.
  - The FCF Project program will use a series of builds to integrate the various components of the system. This allows for progress to be measured and demonstrated as more capabilities are added to the baselines.
  - The testing processes described in this document, up through SQT, will be used on each of the components as they are approved for delivery and test. The SQT test will be used to validate the entire systems performance.

# Version Control

**PVCS Version 6.7**

- Software Version Control based on Capability Maturity Model (CMM) (Level 3)
  - Identification
    - Assign a unique designator to project identified Computer Software Configuration Items (CSCI) and technical data that includes identification of the associated baselines
    - Verify project identification for CSCIs and technical data
    - Assign tracking number to change requests
    - Establish libraries for software and documents
  - Control
    - Receive and place CSCI and technical data in the libraries, thereby providing physical control
    - Process CSCI/technical data requests
    - Provide change request data
    - Deliver software releases from controlled CSCIs and technical data, including associated changes to authorized baselines thus ensuring data integrity

## Version Control – Continued

- Status Accounting
  - Receive Computer Software Configuration Item (CSCI) and technical data for entry into the Configuration Status Accounting (CSA) system (i.e. Data Entry)
  - Generate CSA reports including metrics and schedule data
- Reviews/Audits
  - Support Software Product Assurance (SPA) audit requests for technical data and CSCI and associated data
  - Perform informal reviews of Software Configuration Management (SCM) tasks, Desktop Procedures, and CSA Reports

# Software Reviews

## Joint Technical Reviews

- The FCF Software Development Team plans and participates in Joint Technical Reviews at locations and dates proposed by FDC and approved by NASA Glenn
- The reviews focus on in-process and final software products, rather than materials generated especially for the review. The reviews have the following objectives:
  - Review evolving software products, review and demonstrate technical solutions; and provide insight on the technical effort
  - Identify risks and issues to be raised at joint management reviews
  - Ensure ongoing communication between acquirer and developer technical personnel

# Software Reviews – Continued

## Joint Management Reviews

- FCF Software Development Team plans and participates in Joint Management Reviews at locations and dates proposed by FDC and approved by NASA Glenn
- These reviews are attended by persons with authority to make cost and schedule decisions
- These reviews keep management informed about project status, directions being taken, technical agreements reached, and overall status of evolving software products
- These reviews are used to resolve issues that could not be resolved at joint technical reviews
- Scheduling of these reviews is maintained within the Master Project Schedule

# Software Formal Inspection

Formal Inspection Process based on Capability Maturity Model (CMM) (Level 3) Peer Review Process

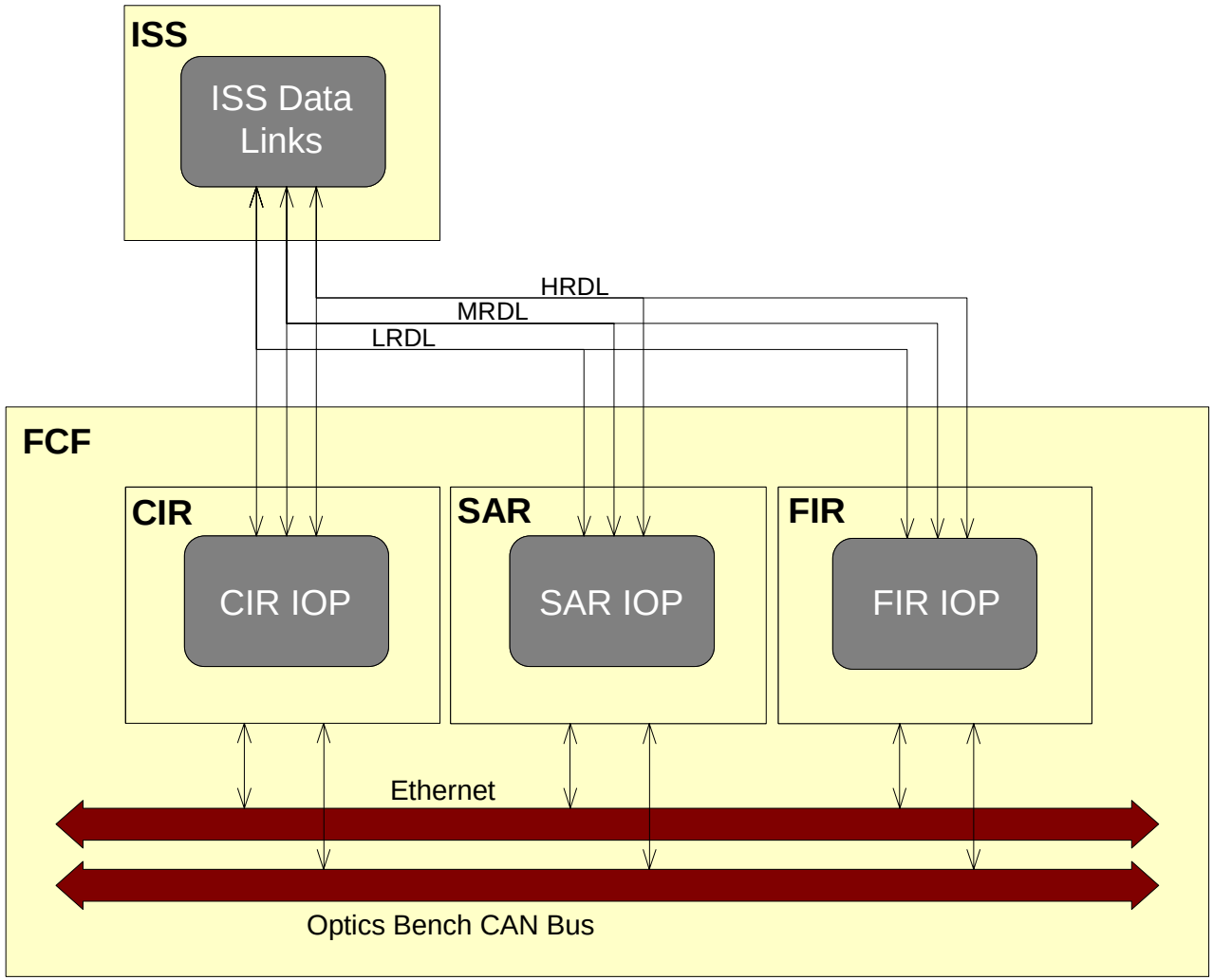
- Software inspections are a rigorous and formalized process that enable the detection and elimination of defects early in the software life cycle.
- The goals of the FDC project software inspections process are to ensure early detection of errors and defects throughout the development process, to ensure that the product is technically correct with respect to the requirements and design specifications, and to collect and analyze inspection data for effective continual improvement of the process and the quality of the product. The software inspection process establishes a foundation for the FDC software metric program.
- FDC developed software products and software documentation are inspected to demonstrate completeness, correctness, and compliance relative to requirements and adherence to project standards.

## Software Formal Inspection – Continued

- FDC developed software products and software documentation are re-inspected if the previously inspected product is significantly modified (i.e., >20%).
- FDC software development products include code (new and significantly modified) and Computer Software Unit (CSU) test plans and procedures. Software documents include items produced during segment development.
- Software inspections are an integral part of the product's developing process and are performed in the early stages of each product's development (e.g., after first draft of a document, clean compile of the code, etc.).
- The author of the product determines when the product is ready for inspection based on pre-inspection checklists.

# FCF Flight Software

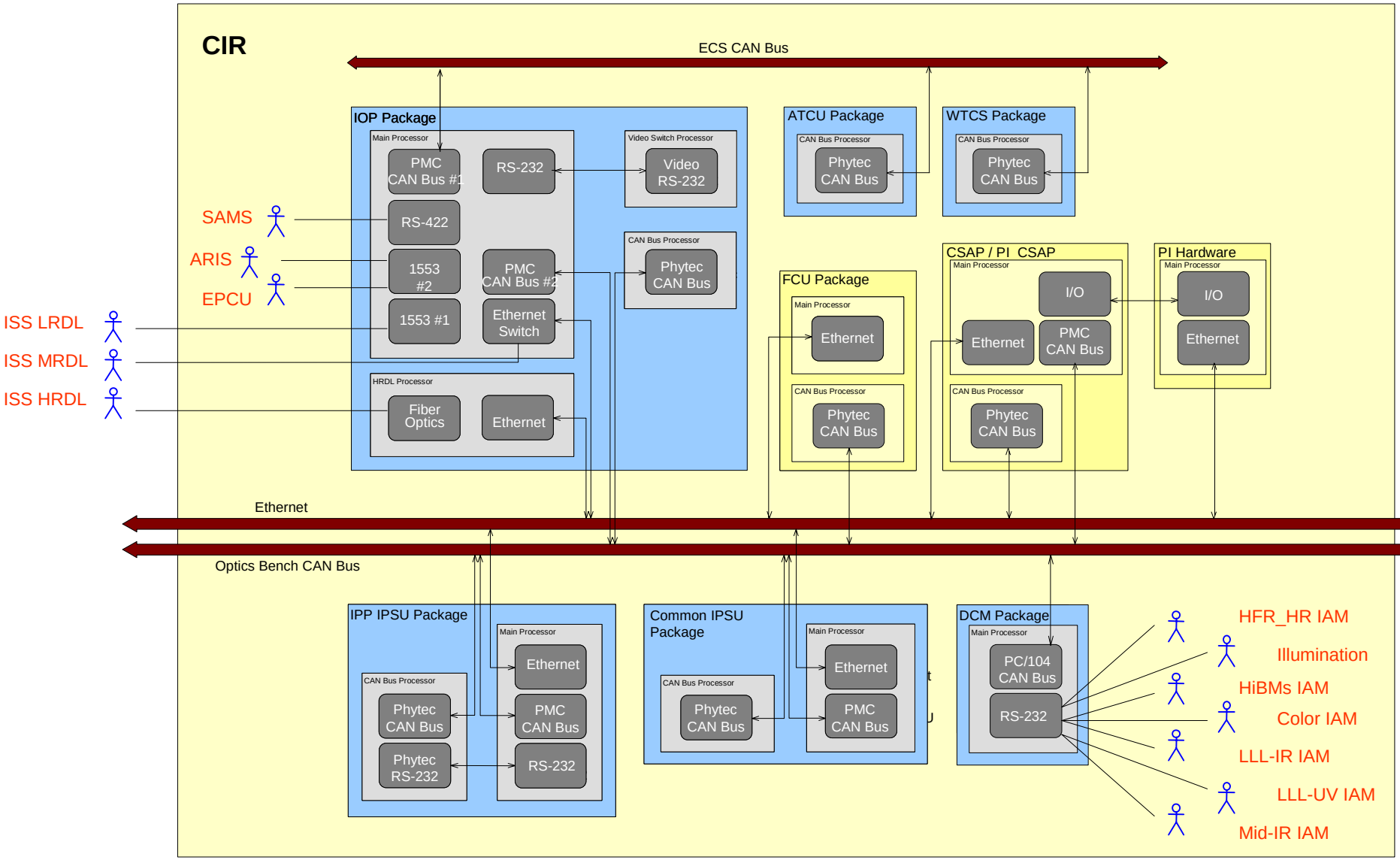
# FCF Systems Architecture



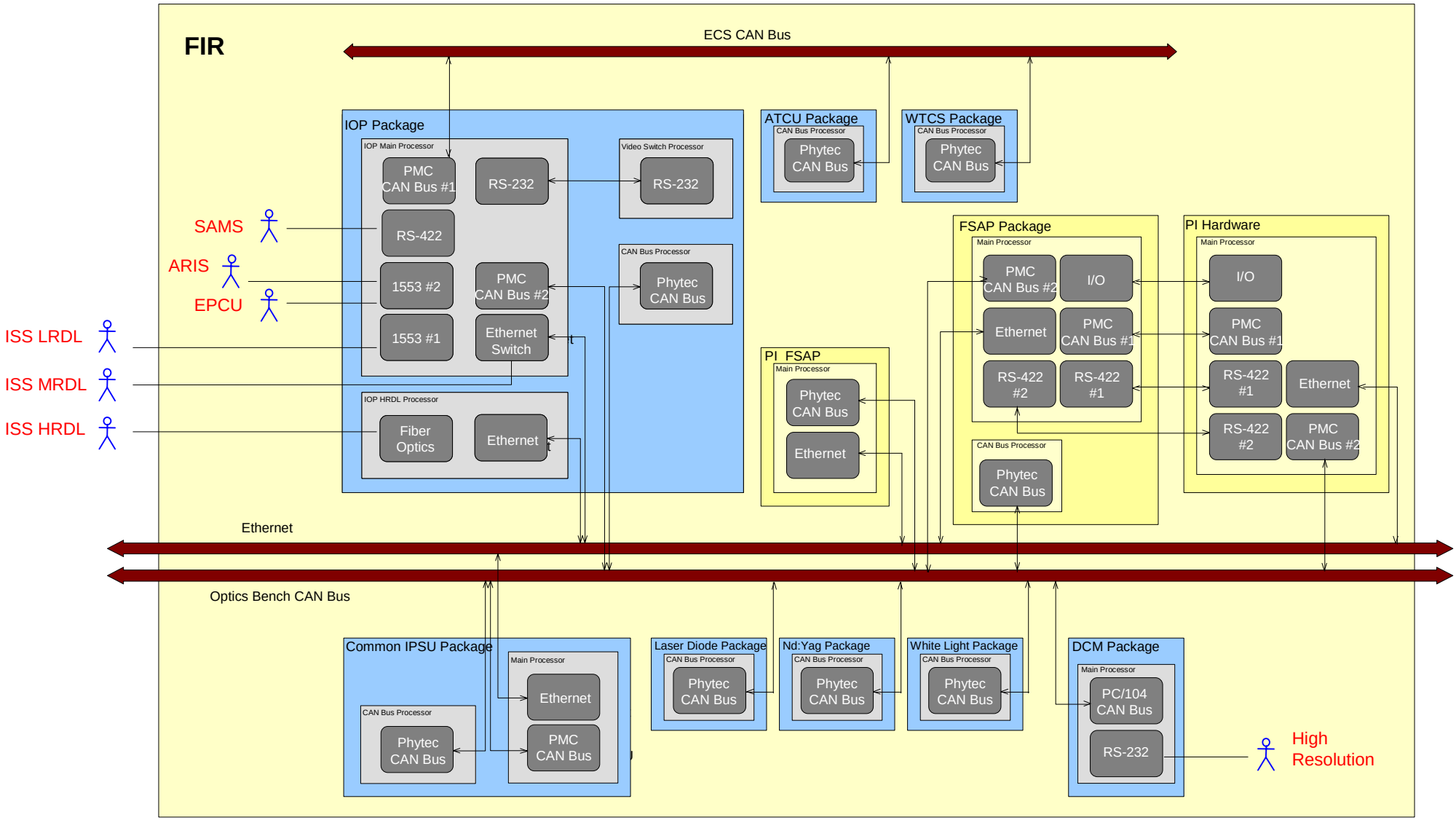
# FCF to ISS Interfaces

- At L-24 the Payload Integration Agreement is signed and Payload Developer defines unique Payload Software requirements and the Command & Data Handling Data Set in Payload Definition Library using SSP 57002 as a guide
  - SSP 57117 Payload Integration Agreement for the Fluids and Combustion Facility
- FCF Rack Interface Details
  - CIR Software Interface Control Document (ICD) with ISS (SSP 57317)
  - FIR Software ICD with ISS (SSP 57318)
  - SAR Software ICD with ISS (SSP 57319)
- Suitcase Test Environment for Payloads (STEP) in FCF Input/Output Processor (IOP) S/W Lab
  - Simulates the ISS's three Data Links
    - Low Rate Data Link (LRDL) – Initial Testing Complete
    - Medium Rate Data Link (MRDL) – In Testing (to date no issues identified)
    - High Rate Data Link (HRDL) – Currently working issues with Boeing, development continues
- Payload Rack Checkout Unit (PRCU) at GRC – Bldg. 333
  - Used for additional Flight Software ISS Verifications

# CIR Systems Architecture

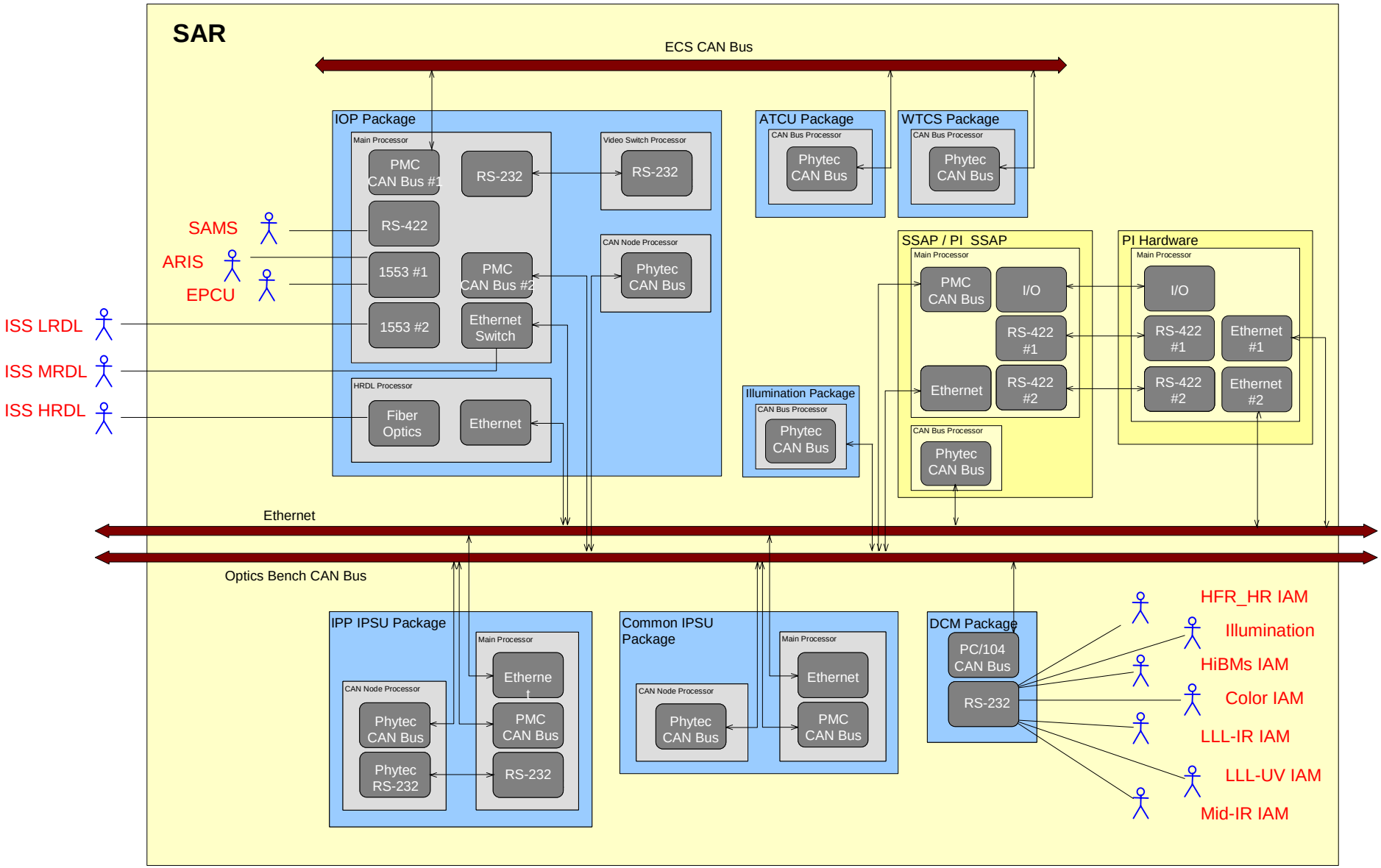


# FIR Systems Architecture



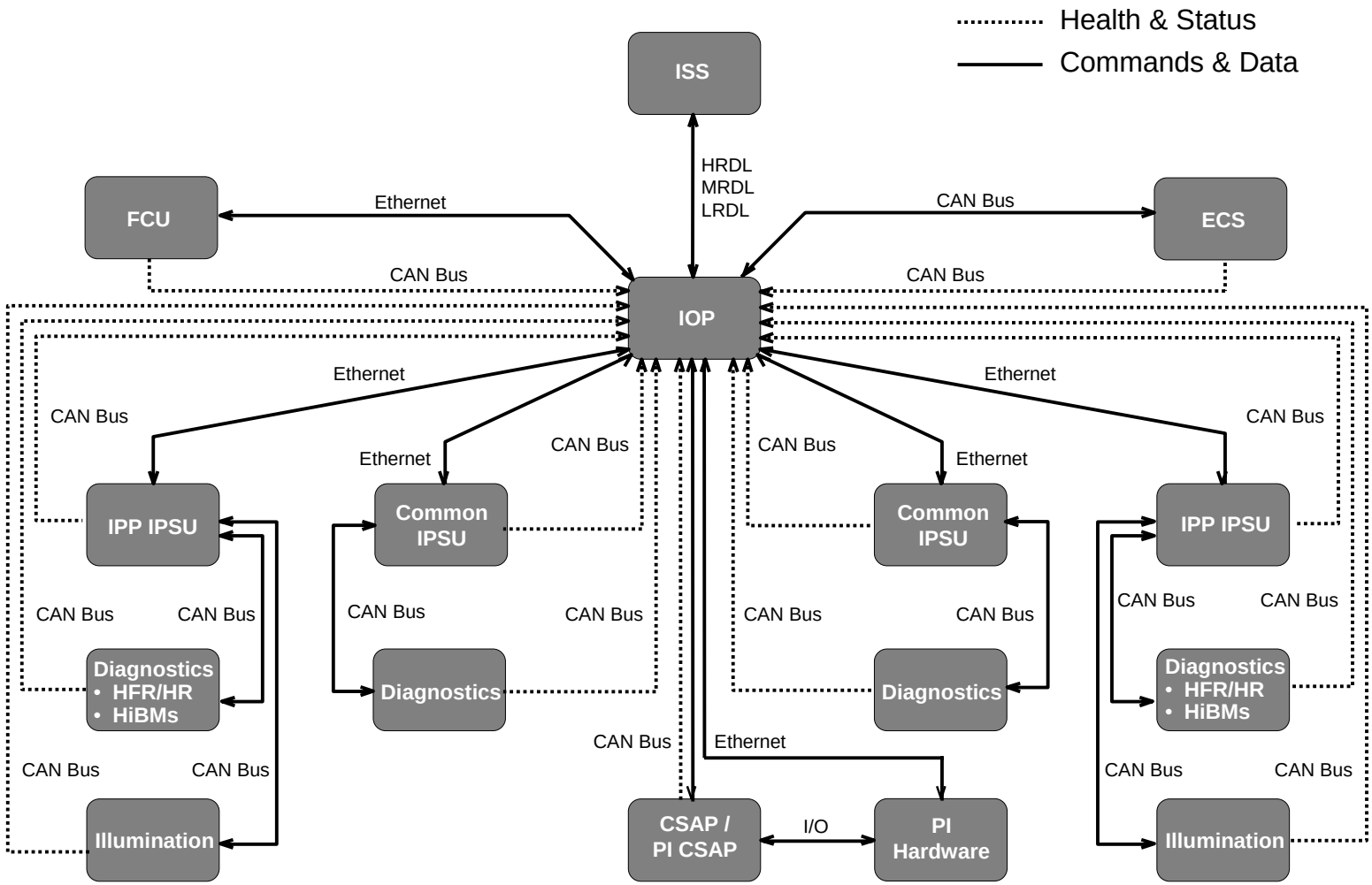

COMMON SOFTWARE FOR CIR, FIR & SAR

# SAR Systems Architecture

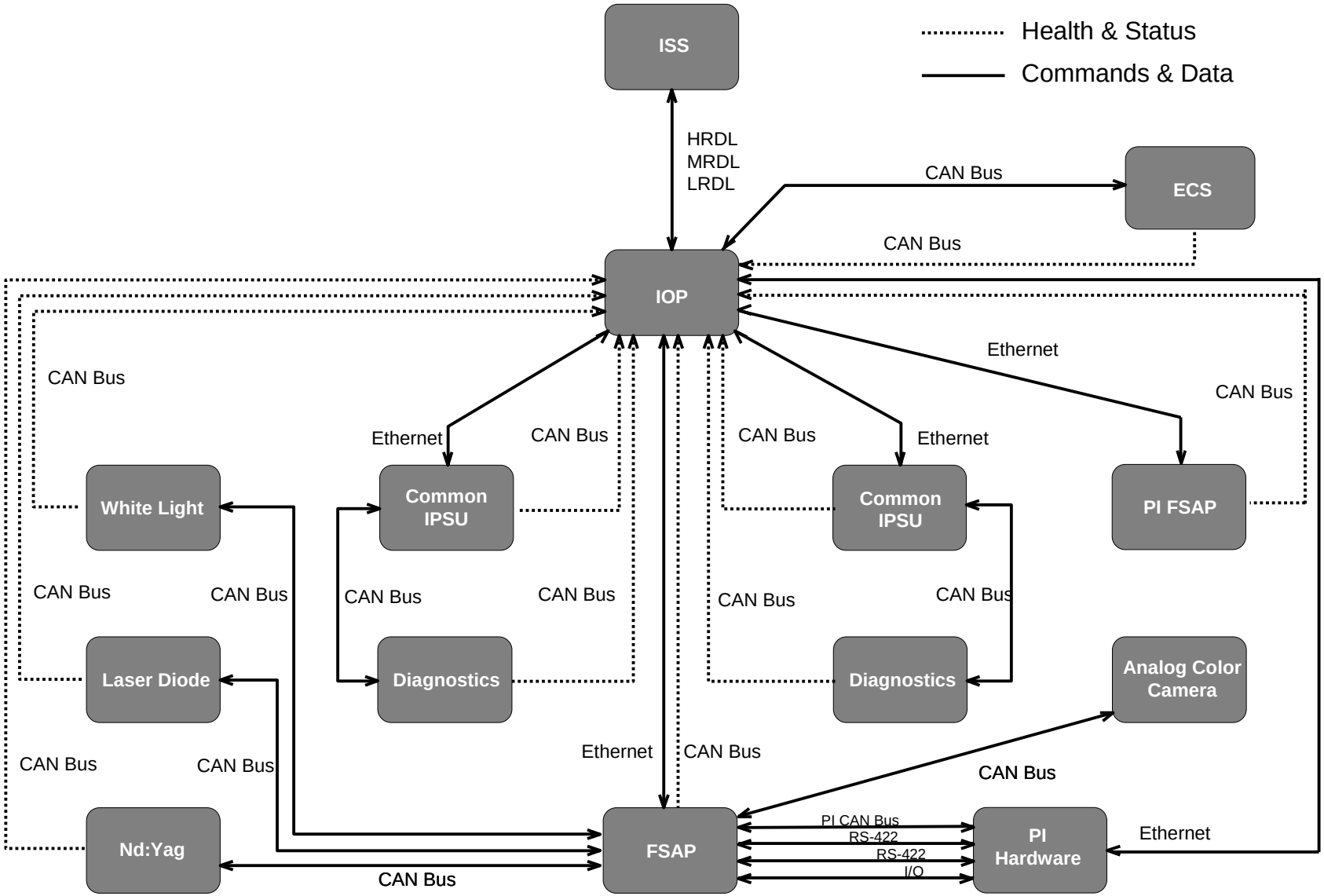



COMMON SOFTWARE FOR CIR, FIR & SAR

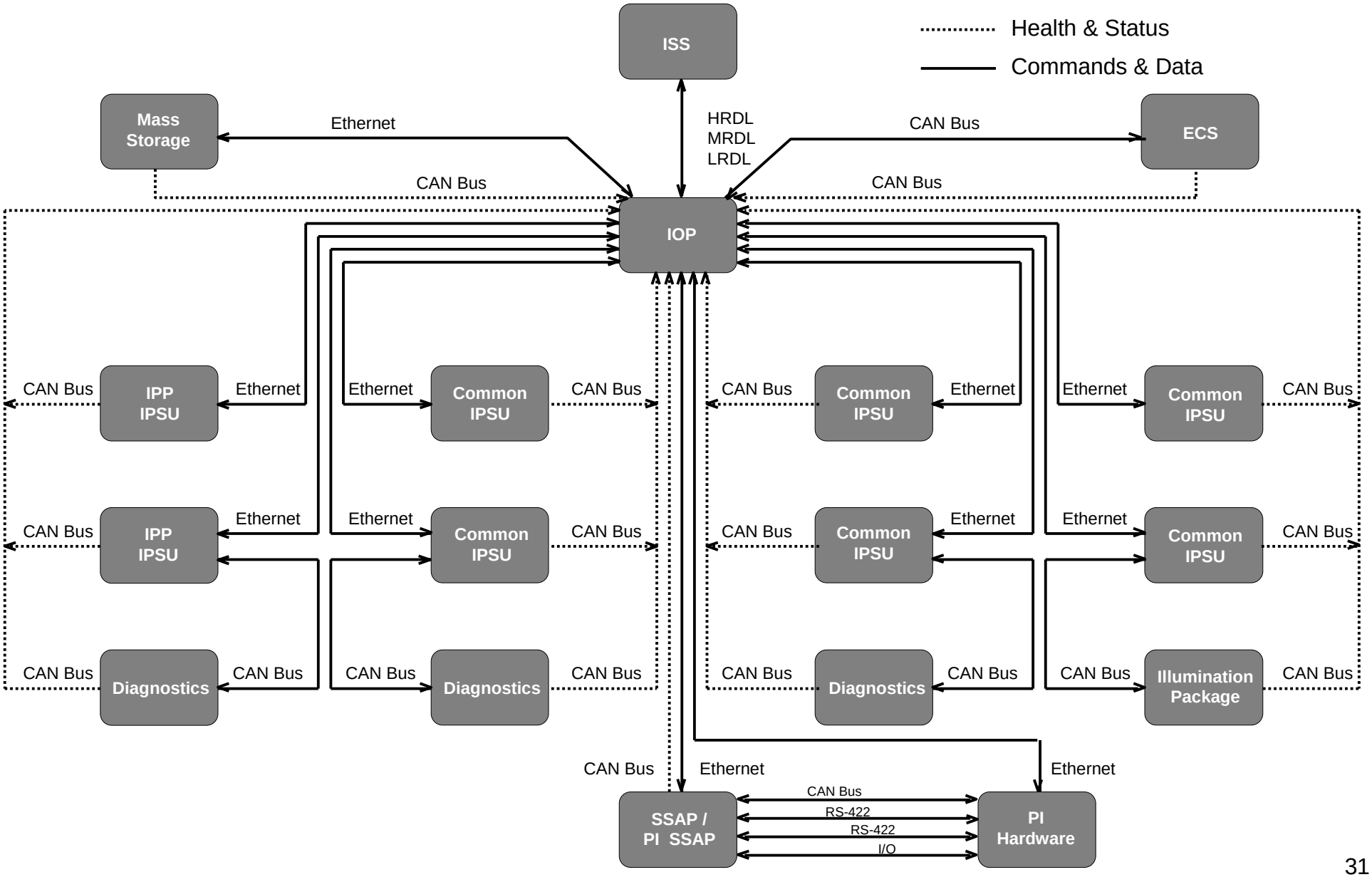
# CIR Nominal Signal Flow



# FIR Nominal Signal Flow



# SAR Nominal Signal Flow



# FCF Software Computer Software Configuration Items (CSCI)

**IOP CSCIs**

- Main Processor
- Video Switch Processor
- High Rate Data Link (HRDL) Processor
- CAN Bus Processor
  - The Input/Output Processor (IOP) provides command and control functions as well as the collection and monitoring of all FCF health and status data. Manages timing functions within the rack and provides the ability for communications between all processors in the FCF. Storage of all science data is accommodated along with the ability to downlink and uplink data and commands through the ISS communications interfaces. Provides ability for communications between all FCF racks.

**ECS CSCIs**

- Air Thermal Control Unit CAN Bus Processor (ATCU CBP)
- Water Thermal Control System CAN Bus Processor (WTCS CBP)
  - The Environmental Control Systems (ECS) provides air thermal and water thermal control for the facilities of the FCF.

# FCF Software CSCIs – Continued

**FOMA Control Unit (FCU) CSCIs**

- Main Processor
- FCU CAN Bus Processor
  - The Fuel/Oxidizer Management Assembly (FOMA) provides the ability to safely deliver all gaseous fuels, diluents, and oxidizers required to perform combustion experiments in the Combustion Chamber. The FOMA can also sample the Chamber environment via a Gas Chromatograph (GC) and control the venting of Chamber gasses, at acceptable concentration levels, to the ISS Vacuum Exhaust System (VES). The FOMA Control Unit (FCU) performs command processing, control, data processing, and health and status monitoring associated with the FOMA.

**FSAP CSCIs**

- Main Processor
- FSAP CAN Bus Processor
  - The Fluids Science Avionics Package (FSAP) is a flexible, multi-purpose data acquisition and control system. It provides a standard set of I/O, controllers, and signal conditioning for the experiment-specific hardware. The FSAP CSCIs will perform command processing, motion, temperature and some specialized scientific devices control, data processing, and status monitoring associated with the experiment. Additionally, it has the ability to control and acquire image data from a camera package, control of illumination packages, as well as motor control for experiment packages. It can allow for science specific capabilities and upgrades.

# FCF Software CSCIs – Continued

**IPP IPSU CSCIs**

- Main Processor
- CAN Bus Processor
  - The Image Processing Package (IPP) Image Processing and Storage Unit (IPSU) CSCIs use ITI hardware to perform the image acquisition, processing, storage, and communication functions necessary to support Illumination Packages and the following Imaging Diagnostic Packages:
    - High Frame Rate/High Resolution (HFR/HR)
    - High Bit Depth Multispectral (HiBMs)

**Common IPSU CSCIs**

- Main Processor
- CAN Bus Processors
  - The Common Image Processing and Storage Unit (IPSU) CSCIs use Digital Signal Processors (DSP) to perform the image acquisition, processing, storage, and communication functions necessary to support Illumination Packages and the following Imaging Diagnostic Packages:
    - Ultra High Frame Rate (UHFR)
    - FIR Standard Camera
    - Low Light Level Ultraviolet (LLL-UV)
    - Low Light Level Near Infrared (LLL-IR)
    - Mid Infrared (MID-IR)

# FCF Software CSCIs – Continued

**DCM CSCI**

- Main Processor
  - The Diagnostic Control Module (DCM) is primarily intended to receive configuration and operational commands from an Image Processing and Storage Unit (IPSU) for command and control of the Diagnostic Packages. It also reports the health and status of those hardware components back to the IPSU and Input/Output Package (IOP).

# Error Detection and Correction

Fault Detection Isolation and Recovery (FDIR) completed at many levels

- POST (Power On Self Test)
  - Board Level Tests to be conducted when power applied to racks and components
  - Identify potential problems prior to experiment start
- 1 Hz Telemetry Checks
  - Telemetry (Data and Commands) monitored for continuity
  - Error messaging sent to ground as faults are identified
- Configuration Verification
  - Ground Software uplinks expected configuration
  - Each rack verifies proper configuration of hardware
  - Ground notified if improper set-up of hardware
- Additional FDIR scenarios are currently being worked by Systems Engineering and when identified will be addressed by the FCF Flight Software Team
  - Single Event Effects (SEE)
  - Hardware Faults at Component & Package Level based on the completed FCF Failure Modes and Effects Analysis (FMEA).

# Payload Software

**Payload Software Interfaces**

- Payload Software Interfaces will be detailed in an FCF to PI Package Software Interface Control Document (ICD)
  - Specifies the Interfaces between the PI Packages and the associated Rack Packages

**Payload Software Capabilities for FCF**

- Payload Software Capabilities will be detailed in the following documentation:
  - FCF Accommodations Handbook (CIR)
  - FCF Accommodations Handbook (FIR)
  - FCF Accommodations Handbook (SAR)
    - These documents detail the capabilities of the Fluids Combustion Facility as they relate to each specific Rack and their associated Payload related requirements

# Status of Software Development

- **Engineering Model (EM) Software**
  - EM Software Requirements developing into Flight Software Requirements
  - Draft Software Design completed
  - Breadboard level software development continues
  - Software testing supports current hardware fidelity
- **Flight Unit Software**
  - ISS Interface Definition continues
  - Baseline Software Management and Development Plan submitted to GRC
  - Preliminary Software Requirements have been submitted to GRC for review
  - Draft Software Design documentation completed. Preliminary Software Design to be submitted by Project CDR
  - Flight Software development at initial stages based on EM software fidelity
- **Ground Software**
  - Ground Operational Scenarios under development
  - Initial stages of Ground Software Requirements definition